robotron 1370 / 1255

Datenerfassungsgeräte
Magnetbandkonverter

Handbuch der Mikroprogrammierung

Handbuch der Mikroprogrammierung

robotron 1370 / 1255

Inhalt:

0. Vorbemerkungen

1. Begriffe der Mikroelektronik/Abkuerzungen/Anglismen

2. Struktur und Funktion der Mikrorechnersysteme

3. Informationsdarstellung im Mikroprozessor

4. Programmierung

5. Befehlsbeschreibung

6. Belegung der Ein- und Ausgabetore

7. Uebersicht ueber die Belegung besonderer Speicherzellen im RAM

8. Funktionszuordnung im ROM

9. Hilfsprogramme bei der Arbeit mit Mikroprogrammen

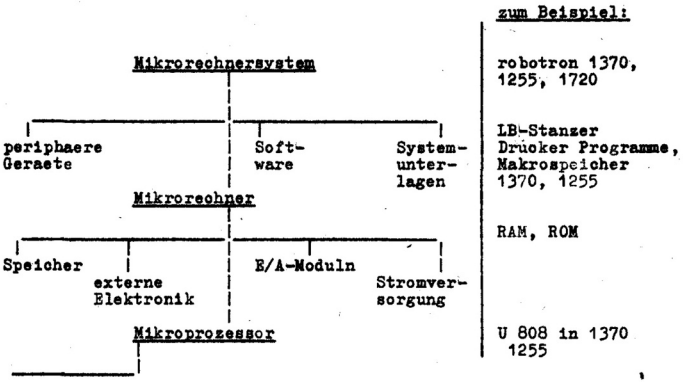
0. Vorbemerkungen

Das "Handbuch der Mikroprogrammierung robotron 1370/1255" soll als Lehrbuch und Nachschlagewerk fuer Programmierer dienen. Technische Einzelheiten sind nur soweit beruecksichtigt, wie sie zum Verstaendnis erforderlich sind und werden teilweise vereinfacht dargestellt.

Der gleichartige Grundaufbau des Datenerfassungsgeraetes robotron 1370 und des Konverters robotron 1255 gestatten es, trotz unterschiedlicher Makroprogrammierung und unterschiedlicher Mikroprogramme ein Handbuch zu nutzen.

1. Begriffe der Mikrorechnertechnik/Abkuerzungen/Anglismen

1.1. Begriffe der Mikrorechnertechnik



- ist eine digitale integrierte Schaltung mit komplexem Schaltungsinhalt
- auf einem Halbleiterchip befinden sich Rechen- und Steuerwerk als Funktionseinheit; man spricht von einer zentralen Verarbeitungseinheit
- ist eine frei programmierbare Recheneinheit mit festem Befehlssatz oder mikroprogrammierbar (z.B. U 808);

1.2. Abkuerzungsverzeichnis

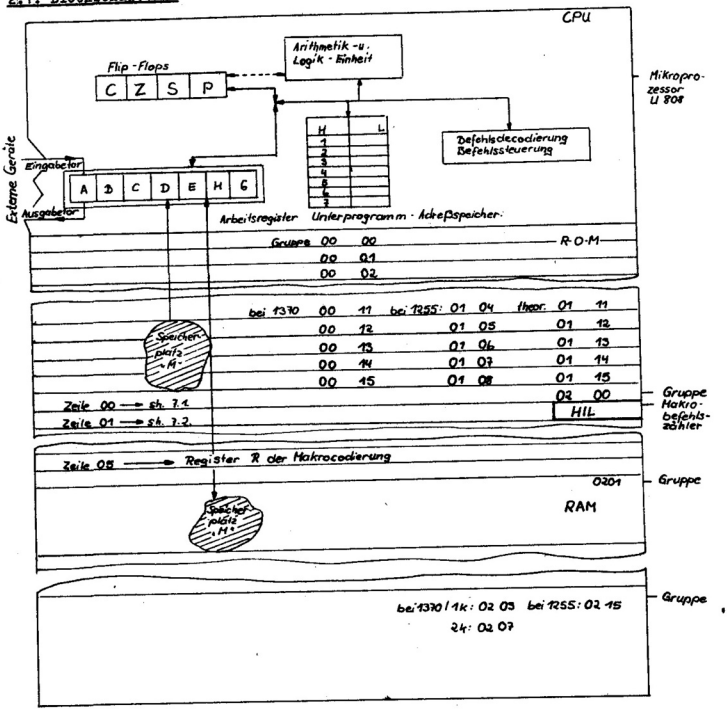
ALE	Arithmetik- und Logik-Einheit
CPU	central-processing-unit (ZVE)
ZVE	Zentrale Verarbeitungseinheit
LSI-Technik	large-scale-integration (Grossintegration)
ROM	Lese-Speicher (read only memory)
RAM	Lese- und Schreibspeicher (random access memory)
E/A-Modul	Ein- und Ausgabemodul
MOS	metal-oxid-semiconductor-Technik
DIL	dual - in - line (Gehaeuseart)
TTL	transistor-transistor logic
FF	Flip-Flop
U 808	Mikroprozessor vom Funkwerk Erfurt

1.3. Anglismen

Pointer-Register	Zeiger-Register
Pins	Kontaktstifte, Zufuehrungsdraehte
arithmetic-logic-unit	Arithmetik-Logik-Einheit
stack and scratch pad memory	dynamischer Stapel-Speicher (Kellerspeicher)
adress stack	Adresstellenspeicher = Unterprogramm - Adressspeicher
scratch pad memory	Arbeitsspeicher (Notizblock-speicher
stack pointer	Stapelzeiger
flop,flag	Flip-Flop
interrupt	Unterbrechung
chip	Bauelement der Mikroelektronik Flaeche ca. 1 cm

2. Struktur und Funktion der Mikrorechnersysteme robotron 1370 und 1255

2.1. Blockschaltbild



2.2. Funktionseinheiten des Mikroprozessors U 808

2.2.1. Ein- und Ausgabetore

- Die (gepufferten) Ein- und Ausgabetore verbinden den Mikroprozessor mit den externen Geraeten.
- Verarbeitungsbreite: 1 Byte = 8 bit.
- Die Bereitstellung der auszugebenden Bitkombinationen (Daten- oder Steuerformationen) erfolgt im Arbeitsregister A.
- Bitkombinationen (Daten- und Steuerinformationen) von externen Geraeten werden im Arbeitsregister A zur Auswertung bereitgestellt.

2.2.2. Befehlscodierung/Befehlssteuerung

- Befehlslaenge: ein Byte,
  zwei Byte (Direktoperationen),
  drei Byte (Sprungbefehle).
- jeder anliegende Befehl wird von der "Befehlscodierung" entschluesselt.
- entsprechend Decodierungsergebnis erfolgt
  - die Steuerung der Prozessablaeufe,
  - das Abrufen des naechsten Befehls.

2.2.3. Befehlszaehler/Unterprogramm-Adresspeicher

- Der Mikroprozessor verfuegt ueber die "echte" Unterprogrammtechnik".
- Unterprogramme koennen maximal siebenmal geschachtelt werden.
- Der Unterprogramm-Adresspeicher enthaelt acht Register:
  - den Befehlszaehler,
  - sieben Adresspeicher zur Aufnahme der Unterprogramm-Rueecksprungadressen
- Der Befehlszaehler erlaubt es, einen 16 K Speicher (ROM und RAM) zu adressieren.
- Der Befehlszaehler speichert die Adresse des (gerade) abzuarbeitenden Befehls. Nach der Befehlsausfuehrung erhaelt der Befehlszaehler die Adresse des naechsten abzuarbeitenden Befehls. Befehlszaehler ist die Adresse des naechsten abzuarbeitenden Befehls.
- Bei Aufruf eines Unterprogramms (CALL-Befehl) rueckt der Befehlszaehler um eins nach unten (siehe Blockschaltbild). Die so erreichte Ebene im Unterprogramm-Adresspeicher dient im Unterprogramm als Befehlszaehler.
  Im "alten" Befehlszaehler wird die Adresse des naechsten im uebergeordneten Programm abzuarbeitenden Befehls gespeichert. Mit Rueckkehr aus dem Unterprogramm (RETURN-Befehl) wird der Befehlszaehler um eins nach oben gerueckt.
  Das Programm wird mit jenem Befehl fortgesetzt, dessen Adresse im erreichten Befehlszaehler gespeichert ist.
- Wird die Kapazitaet des Speichers ueberschritten (mehr als sieben geschachtelte Unterprogramm-Aufrufe), so werden entsprechende Speicherebenen zerstoert; das Programm wird undefiniert fortgesetzt.

2.2.4. Arbeitsregister

- Der Mikroprozessor verfuegt ueber sieben Arbeitsregister á 1 Byte (= Verarbeitungsbreite).

  Bezeichnungen: A, B, C, D, E, H, L.

- Die Arbeitsregister bilden quasi den "internen Speicher" des Mikroprozessors.

- Einige Register erfuellen Sonderfunktionen.

2.2.4.1. Register A (Akkumulator)

- Register A erfuellt im Mikroprozessor die Akkumulatorfunktion.
- Register A enthaelt bei allen logischen und arithmetischen Operationen den ersten Operanden.
  Das Ergebnis dieser Operationen steht in A (Einadressprinzip).

2.2.4.2. Register H und L

- In H (Hoeherer Adressteil = Gruppe) und L (niederer Adressteil = Byteadresse) wird bei Operationen, die mit dem ROM- oder RAM-Speicher zusammenwirken, die Adresse eines Bytes gespeichert.
- Der Speicherplatz, dessen Adresse in H und L zwischengespeichert ist, wird allgemein als "M" bezeichnet.
- M ist quasi ein "externes" Register der Laenge 1 Byte, mit dem der Mikroprozessor zusammenwirken kann (ROM-Bereich nur lesend!).
- H und L koennen auch als Arbeitsspeicher genutzt werden.

2.2.4.3. Register B, C, D, E

- B, C, D und E sind Arbeitsregister, die vom Programmierer beliebig verwendet werden koennen.
- In den Registern B und C wird ausserdem nach Ausfuehrungen des Makrobefehls "Sprung ins Mikroprogramm" bei robotron 1370/1255 die Adresse des naechsten im Makroprogramm abzuarbeitenden Befehls im Register zwischengespeichert.

2.2.5. Arithmetik- und Logikeinheit

- Die logischen und arithmetischen Operationen werden von einem 8-bit-Binaerrechnerwerk mit paralleler Uebertragsbildung ausgefuehrt.
- Abhaengig vom Ergebnis der arithmetischen/logischen/Verschiebe - Operationen werden von der ALE bis zu vier Kennzeichen in sogenannten Flip-Flops eingestellt.

2.2.6. Bedingungen des Mikroprozessors fuer die Programmverzweigung

Die als Ergebnis der Operationen eingestellten 1..4 Kennzeichen bilden im Befehlsvorrat des Mikroprozessors die Bedingungen fuer die Programmverzweigung.
Die Kennzeichen koennen den Wert 0 oder 1 haben.
Sie koennen einzeln ausgewertet und damit fuer die Programmverzweigung genutzt werden.

2.2.6.1. Kennzeichen C (Uebertrags-FF)

- C <--- carry (Uebertrag)

- C wird 1 gesetzt, wenn ein Ueberlauf aus den hoechstwertigen Bit des Arbeitsregisters A auftritt. (siehe Zeichendarstellung, Pkt. 3.1.).

2.2.6.2. Kennzeichen Z (Null-FF)

- Z <--- zero (Null)

- Z wird bei einer logischen oder arithmetischen Operation 1 gesetzt, wenn das Resultat gleich Null ist.

2.2.6.3. Kennzeichen S (Vorzeichen-FF)

- S <--- sign (Vorzeichen)

- S wird bei einer logischen oder arithmetischen Operation 1 gesetzt, wenn das hoechste bit gleich 1 ist (siehe Zeichendarstellung Pkt. 3.1.).

2.2.6.4. Kennzeichen P (Paritaets-FF)

- P <--- Parity

- P wird bei gerader Anzahl von Null verschiedener Bitstellen (2, 4, 6 oder 8 Bitstellen) nach arithmetischen oder logischen Operationen gleich 1 gesetzt.

3. Informationsdarstellung im Mikroprozessor

Alle Informationen des Mikroprozessors werden d u a l in einem Datenwort fester Laenge dargestellt

8 bit = 1 Byte.

Innerhalb des Mikroprozessors werden die Informationen von einem

8-bit-Datenbus

uebertragen.

Fuer die Uebertragung von Speicheradressen nach den Registern H und L (vgl. 2.2.4.2.) bzw. nach dem Adresspeicher des Mikroprozessors sind deshalb zwei aufeinanderfolgende Uebertragungen erforderlich.

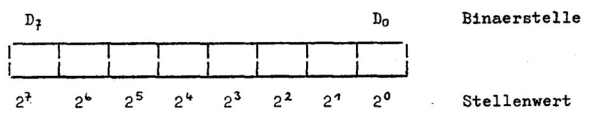
Die Verbindung des Mikroprozessors zu den angeschlossenen Speichern und externen Geraeten wird durch einen

E/A-Puffer

hergestellt.
Auch der E/A-Puffer hat eine Laenge von 8 bit
(Anschluesse D_0 ... D_7).

Ueber diesen Puffer erfolgt der gesamte Befehls- und Datenaustausch.

Datenwort:

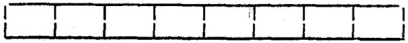


Das Datenwort kann demnach intern die Werte 0...255 annehmen.

3.1. Zahlendarstellung

Die interne Darstellung kann durch Vereinbarung der Lage eines Kommas unterschiedlich interpretiert werden:

Ganze Zahl (positiv)



$0 \leq x \leq 255$

Festkommazahl (einfache Genauigkeit)

2^{-1} 2^{-2} 2^{-3} 2^{-4} 2^{-5} 2^{-6} 2^{-7} 2^{-8} $0 = x = 0{,}99609375$

Muessen Zahlen groesserer Laenge verarbeitet werden, wird dazu intern das Uebertragungszeichen C (vgl. 2.2.6.1.) genutzt:

Mehrbytaddition (Beispiel)

	Byte n	...	Byte 2	Byte 1
Zahl x	x_n	...	x_2	x_1
Zahl y	y_n	...	y_2	y_1
Uebertrag C	C_{n-1}	...	C_1	-
x + y = z	z_n	...	z_2	z_1

Negative Zahlen werden durch das Zweierkomplement dargestellt. Die Stelle D_7 wird als Vorzeichen gedeutet:

$D_7 = 0$ positive Zahl

$D_7 = 1$ negative Zahl

Vom Rechnerwerk wird D als "normale" Binaerstelle behandelt.

Als Zahlenbereich ergibt sich infolge der Vorzeichendarstellung in D_7

$$-2^7 = x = 2^7 - 1$$

Fuer die Bildung des Zweierkomplements gibt es zwei Moeglichkeiten:

1. Invertierung jeder Binaerstelle und Addition von 1

2. Pruefung aller Binaerstellen ab D_0 auf $\neq 0$,

Invertierung aller Binaerstellen links von der ersten von 0 verschiedenen Stelle

Anmerkung:

Invertierung ist die Umkehrung des Zustandes jeder Bitstelle (0 --> 1, 1 --> 0).

Durch die Bildung des Zweierkomplements wird die Subtraktion im Mikroprozessor auf eine Addition zurueckgefuehrt und nur so ausgefuehrt.

Beachte:

Das Uebertragungskennzeichen C ist unterschiedlich

a) nach der Addition von Komplementzahlen bzw.
b) nach der Subtraktion "normaler" Zahlen,

obwohl die 8 Stellen des Datenwortes das gleiche Ergebnis zeigen:

1. Bildung einer Komplementzahl

```
+ 12     0 0 0 0 L L 0 0

         L L L L 0 0 L L
       + 0 0 0 0 0 0 0 L
       ------------------
         L L L L 0 L 0 0        Komplement von 12
       ==================
```

2a. Addition der Komplementzahl

```
     + 10    0 0 0 0 L 0 L 0
  +  (- 12)  L L L L L L L 0
     ----------------------
     C = 0   L L L L L L L 0
     ======================
```

2b. Subtraktion der normalen Zahl

```
    + (-10)   0 0 0 0 L 0 L 0
  + (+ 12)    0 0 0 0 L L 0 0
  ----------------------------
    0 = L     L L L L L L L 0
  ============================
```

Ergebnis: $(D_7 \ldots D_0)_a = (D_7 \ldots D_0)_b$

$C_a \neq C_b$

3.2. Befehlsdarstellung

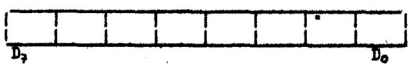
Der Mikroprozessor wird durch die intern entschluesselten Mikrobefehle sowie durch interne Signale gesteuert.

Ein Mikrobefehl enthaelt die Art der auszufuehrenden Operationen und die Operanden-Adressen oder bei Programmverzweigungen die Befehle-Adressen fuer die auszufuehrende Sprungoperation.

Je nach Art der auszufuehrenden Operation haben die Mikrobefehle eine Laenge von

1, 2 oder 3 Byte:

1-Byte-Befehle

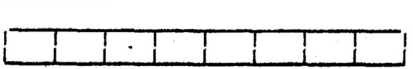


sind dann moeglich, wenn

a) eine Register-Adresse (vgl. 2.2.4.) angegeben werden muss (3 bit). Dann sind 5 bit fuer den Operationscode verfuegbar, oder

b) zwei Register-Adressen angegeben werden muessen (6 bit) und zwei bit fuer den Operationscode ausreichen.

2-Byte-Befehle

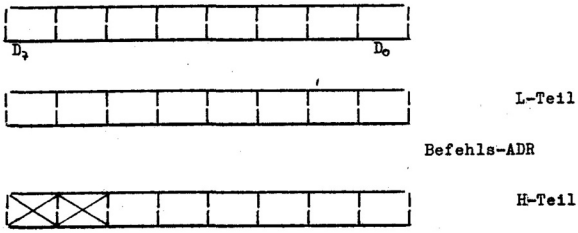


werden benoetigt, wenn

im ersten Byte der Operationscode (≤ 5 bit) und eine Register-Adresse (3 bit) angegeben ist und

im zweiten Byte ein beliebiger Wert als sogenannter Direktoperand (vgl. 3.3.3.) angegeben ist.

3-Byte-Befehle



werden benoetigt, wenn bei Programmverzweigungen.

Im **ersten** Byte wird die Art der Sprungoperation, im **zweiten** Byte der niedere Adressteil (L) und im **dritten** Byte der hohe Adressteil (H) des Sprungziels angegeben.

Bei ausgefuehrtem Sprung werden die 14 bit der neuen Befehls-Adresse in den Adresspeicher des Mikroprozessors in die Byte L (8 bit) und H (6 bit) entsprechend eingeschrieben.

3.3. Adressierungsarten

Bei den Mikroprozessoren ist die direkte und die indirekte Adressierung moeglich.
Ausserdem wird die unmittelbare Definition von Operanden im Befehl (Direktoperanden) angewendet.

3.3.1. Direkte Adressierung

Die direkte Adressierung wird bei den Arbeitsregistern A...L sowie bei Programmverzweigungen angewendet.

Fuer die Adressierung der Arbeitsregister werden 3 bit benoetigt:

A	0	0	0
B	0	0	L
C	0	L	0
D	0	L	L
E	L	0	0
H	L	0	L
L	L	L	0

Fuer die komplette Adressierung des Speichers werden 14 bit benoetigt (Sprung an jedes Byte des Speichers ist damit prinzipiell moeglich).

3.3.2. Indirekte Adressierung

Die endgueltige Adresse fuer die Ausfuehrung einer Operation wird durch Modifikation einer vorgegebenen Adresse ermittelt. Dafuer kennt der Mikroprozessor mehrere Moeglichkeiten:

3.3.2.1. Indirekte Adressierung ueber Speicheradress-Register

Wie unter 3.3.1. dargestellt, werden fuer die komplette Adressierung des Speichers 14 bit benoetigt.
Das gilt fuer die direkte Darstellung der Sprung-Adressen ebenso wie fuer die Adressierung des Speichers mit dem Ziel arithmetischer oder logischer Operationen.

Die Adresse des Operanden mit der allgemeinen Bezeichnung "M" wird in die Speicheradress-Register L (8 bit fuer niederen Teil) und H (6 bit fuer den hoeheren Teil) eingeschrieben.
Die Ausfuehrung der Operation erfolgt in zwei Schritten:

1. Laden der Speicheradress-Register mit der Adresse des Operanden "M"

2. Ausfuehrung der Operation mit "M".

3.3.2.2. Adressensubstitution

Die Substitution von Adressen kann ebenfalls nur ueber die Speicheradress-Register H und L erfolgen:

1. Lade Register H, L mit Operanden-ADR "M"
   (M selbst enthaelt Adressdaten!)

2. Laden H und/oder L mit M

3. Ausfuehrung der Operation mit M.

3.3.2.3. Indizierte Adressierung

Der Mikroprozessor verwendet fuer die Indizierung keine speziellen Befehle. Ausser dem Arbeitsregister A koennen aber praktisch alle Arbeitsregister als Indexregister benutzt werden, da direkte Befehle fuer Addition bzw. Subtraktion von 1 vorhanden sind.
Wird die Basis-Adresse in die Speicheradress-Register H und L geladen, kann die Aktualisierung des als Index eingesetzten Registers direkt erfolgen (z.B. "L" - Weiterzaehlen um $\pm$ 1).

3.3.3. Definition direkter Operanden

Der Mikroprozessor kennt die Moeglichkeit, unter Umgehung einer gesonderten Anfangswertbildung und damit verbundener Adressierung direkt im Befehl einen Operanden zu definieren.

Bei entsprechendem Operationscode wird das zweite Byte des Befehls als Datenbyte erkannt und direkt fuer die Operation verwendet.

Die Operation kann sowohl mit einem Arbeitsregister (direkte Adressierung) als auch mit dem durch H und L definierten Speicherplatz M (indirekte Adressierung) erfolgen.

4. Programmierung robotron 1370/1255

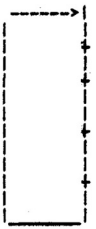
Die Verarbeitungsbreite (8 bit) und die Adressierungsmoeglichkeit (14 bit) des Mikroprozessors U 808 bedingen eine elementare Gestaltung des Befehlsvorrates. Diese Tatsache ist vorteilhaft fuer die Anwendung des Mikroprozessors in einem Mikrorechner. Aus den elementaren Mikrobefehlen des Mikroprozessors lassen sich die gewuenschten Funktionen des Mikrorechners zusammensetzen. Jede dieser Funktionen (Makrobefehle des Mikrorechners) loest demnach eine festprogrammierte Folge von Mikrobefehlen aus. Diese Mikrobefehlsfolgen und -zyklen sind im ROM gespeichert. In den Makrobefehlsvorrat werden solche Befehle aufgenommen, die den anwendungstechnischen Forderungen des Mikrorechnersystems entsprechen.
Fuer den Fall robotron 1370 bedeutet es, dass die Makrobefehlsliste fuer Probleme der Datenerfassung aufgestellt wurde und die fuer robotron 1255 zur Loesung von Aufgaben der Konvertierung.

Der Ablauf der Arbeiten an den Anlagen robotron 1370/1255 ist festgelegt durch die im RAM befindlichen Anwendungsprogramme. Diese Programme bestehen im allgemeinen aus Makrobefehlen. Der Anwender kann Standardprogramme oder auch selbst erarbeitete Programme benutzen.

4.1. Funktion des Mikroprogrammes im ROM

Nach dem Start des Anwenderprogrammes im RAM uebernimmt das Anwenderprogramm gemaess der darin programmierten Befehle die Ablaufsteuerung der Arbeiten an den Anlagen robotron 1370/1255. Jeder Makrobefehl wird durch fest zugeordnete Mikrobefehlsfolgen und -zyklen verwirklicht. Die logische Verarbeitung der Befehle nacheinander (linear oder verzweigt durch Sprungbefehle) fuehrt der spezielle Zyklus "Makrodecodierung" durch. Dieser Mikroprogrammzyklus ist ebenfalls im ROM enthalten.

Makrodecodierung (vereinfachtes Schema)

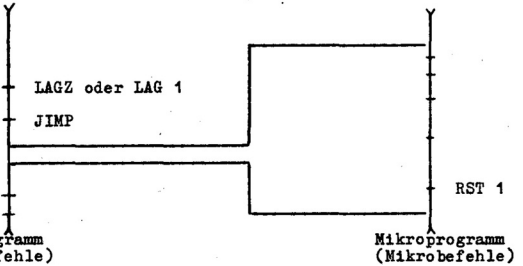


1) Dabei beruecksichtigt die Makrodecodierung die unterschiedlichen Befehlslaengen (Ein- und Zweibytebefehle bei robotron 1370, Ein-, Zwei- und Dreibytebefehle bei robotron 1255).
Die Durchfuehrung des Makrobefehls geschieht in den entsprechenden Befehlsfolgen und -zyklen, die im ROM fest programmiert vorhanden sind. Am Ende jeder Ausfuehrung eines Makrobefehls wird im allgemeinen an den Anfang der Makrodecodierung zurueckgesprungen und der logisch naechstfolgende Makrobefehl abgearbeitet. Nach den Befehlen "Sprung ins Vorwahlprogramm" und "Sprung ins Mikroprogramm" erfolgt keine Fortsetzung in der Makrodecodierung.

4.2. Zusammenwirkung der Makro- und Mikroprogrammierung im RAM

Die Anlagen robotron 1370/1255 geben dem Anwender die Moeglichkeit, Programme abzuarbeiten, die aus Mikrobefehlen des Mikroprozessors geschrieben wurden. Dazu muessen diese Mikroprogramme in den RAM eingelagert werden. Der Ansprung dieser Mikroprogramme erfolgt mit Hilfe des Makrobefehls "Sprung ins Mikroprogramm" - JIMP.
Nach der Abarbeitung dieses Makrobefehls kehrt man nicht zur Makrodecodierung des naechsten Makrobefehls zurueck, sondern es wird die Abarbeitung der angesprungenen Mikrobefehle begonnen.
Der Makrobefehlszaehler enthaelt nach dem Absprung die Adresse des auf dem Makrobefehls "JIMP" nachfolgenden Byte.
Die Adressierung der abzuarbeitenden Mikrobefehle ist in den Anlagen robotron 1370/1255 unterschiedlich vorzunehmen.

Uebergang vom Makro- zum Mikroprogramm in der Anlage robotron 1370 im RAM-Bereich:



Vor dem Befehl "Sprung ins Mikroprogramm" muss mit Hilfe des Befehls "Laden Indexregister mit Adresse Gruppe 0" oder "Laden Indexregister mit Gruppe 1" in das Indexregister 3 die Gruppenadresse des RAM-Platzes geladen werden, auf dem der erste Mikrobefehl gespeichert ist. Die zugehoehrige Byteadresse steht im zweiten Byte des Sprungs "JIMP".

Codierung der Befehle: LAGZ 09 11 xx xx oder
LAG 1 09 15 xx xx

(xx xx = Gruppenadresse 02 00, 02 01, 02 02, 02 03)
und bei 2 K-Byte-RAM auch
02 04, 02 05, 02 06, 02 07
JIMP 13 15 yy yy

(yy yy = Byteadresse von 00 00 bis 15 15)

Die Befehle LAGZ bzw. LAG 1 und JIMP muessen nicht unmittelbar aufeinanderfolgen. Es koennen andere Mikrobefehle dazwischen stehen, sofoern sie nicht das Indexregister 3 beeinflussen.

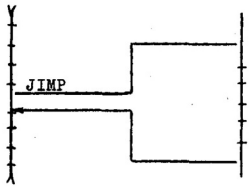
Der Mikrobefehl

RST 1, Codierung 00 13

bewirkt die Rueckkehr ins Makroprogramm.

Er gehoert zur Gruppe der Unterprogrammspruenge mit fester Adresse und ruft das Mikroprogramm "Makrodecodierung" auf. Da zum Zeitpunkt des Ueberganges in das Mikroprogramm der Makrobefehlszaehler die Adresse desjenigen Bytes enthaelt, das auf den Sprung "JIMP" unmittelbar folgt, wird das Makroprogramm auch an jener Stelle fortgesetzt. Eine Abweichung von dieser Fortsetzungsart ist moeglich, wenn in dem durch "JIMP" augerufenen Mikroprogramm der Inhalt des Makrobefehlszaehlers veraendert wurde.

Uebergang vom Makro- zum Mikroprogramm in der Anlage robotron 1255 im RAM-Bereich:



Da in der Anlage robotron 1255 der Befehl

"Sprung ins Mikroprogramm"

ein Dreibytebefehl ist, kann die komplette Adresse des anzuspringenden Mikroprogramms in dem Sprungbefehl "JIMP" codiert werden:

```
JIMP  12  15  xx  xx  yy  yy
              xx  xx  =   Gruppenadresse von 02 00 bis 02 15
              yy  yy  =   Byteadresse    von 00 00 bis 15 15
```

Die Rueckkehr in das Makroprogramm geschieht auf die gleiche Art wie bei der Anlage robotron 1370.
Der Sprung "JIMP" gestattet es, in den Anlagen robotron 1370 und 1255 auch Mikroprogramme im ROM-Bereich anzuspringen. Erlaeuterungen dazu stehen unter Punkt 8.

5. Befehlsbeschreibung

5.1. Allgemeine Hinweise

Die Befehlscodierung ist aus der Spalte Operationscode aus der entsprechenden Matrix direkt waehlbar.

5.2. Symbolerlaeuterung

A	Register	A	(Akkumulator)
B	"	B	
C	"	C	
D	"	D	
E	"	E	
H	"	H	(Speicheradressregister)
L	"	L	(Speicheradressregister)

M wird durch H und L dargestellt
q feste Adresse
c steht fuer den Inhalt der Bedingungs-Flip-Flop C/Z/S/P
x Operationsbodes beliebig; Vorzugscode bei x = 0
r Register allgemein
r_1 Zielregister (alter Inhalt wird verdraengt)
r_2 Quellregister (Inhalt bleibt erhalten)
n beliebiger Wert (Direktoperand)
ladr niedere 8 Bit der Adresse
nadr hohe 6 Bit der Adresse
C Ueberrags-FF; 1 bei Uebertrag
Z Null-FF; 1 bei Null
S Vorzeichen-FF; 1 bei Vorzeichenstelle = 1
P Paarigkeits-FF, 1 bei paarig.

5.3. Erlaeuterungen zur Darstellung der Ein- und Ausgabetore

- gestrichene Felder sind nicht belegt
- offene Felder nur fuer Techniker von Bedeutung

lfd. Nr.			Operationscodes r	0	1	2	3	4	5	6	7	2.Byte	3.Byte
	Ladebefehle		r	A	B	C	D	E	H	L	M		
			A	1200	1201	1202	1203	1204	1205	1206	1207		
			B	1208	1209	1210	1211	1212	1213	1214	1215		
1	L r r	r := r	C	1300	1301	1302	1303	1304	1305	1306	1307		
2	L r M	r := M	D	1308	1309	1310	1311	1312	1313	1314	1315		
3	L M r	M := r	E	1400	1401	1402	1403	1404	1405	1406	1407		
			H	1408	1409	1410	1411	1412	1413	1414	1415		
			L	1500	1501	1502	1503	1504	1505	1506	1507		
			M	1508	1509	1510	1511	1512	1513	1514	x		
4	L r n	r := n		0006	0014	0106	0114	0206	0214	0306		n	
5	L M n	M := n									0314	n	
	Inkrement-u.Dekrementbefehle												
6	I N r	r := = r + 1.		A	B	C	D	E	H	L			
8	D C r	r := = r - 1	r:	x	0008	0100	0108	0200	0208	0300			
				x	0009	0101	0109	0201	0209	0301			
	Additions-u. Subtraktionsbefehle												
			r:	A	B	C	D	E	H	L	M		
9	A D r	r		0800	0801	0802	0803	0804	0805	0806			
10	D D M	A := A + M									0807		
11	A D n	n		0004								n	
12	A C r	r		0808	0809	0810	0811	0812	0813	0814			
13	A C M	A := A + M + C									0815		
14	A C n	M		0012								n	
15	S U r	r		0900	0901	0902	0903	0904	0905	0906			
16	S U M	A := A - M									0907		
17	S U n	n		0104								n	
18	S B r	r		0908	0909	0910	0911	0912	0913	0914			
19	S B M	A := A - M - e									0915		
20	S B n	n		0112								n	
	Vergleich												
			r:	A	B	C	D	E	H	L	M		
21	CP r	r r		1108	1109	1110	1111	1112	1113	1114			
22	CP M	C := A < M;Z:=A= M									1115		
23	CP n	n n		0312								n	

lfd.					Operationscodes								2.Byte	3.Byte
	Logische Operationen (UND; Ex,ODER;ODER)				r: A	B	C	D	E	H	L	M		
24	ND r			r	1000	1001	1002	1003	1004	1005	1006			
25	ND M	A	:=	A M								1007		
26	ND n			n	0204								n	
27	XR r			r	1008	1009	1010	1011	1012	1013	1014			
28	XR M	A	:=	A M								1015		
29	XR n			n	0212								n	
30	OR r			r	1100	1101	1102	1103	1104	1105	1106			
31	OR M	A	:=	A M								1107		
32	OR n			n	0304								n	
	Rotationsbefehle (links/rechts)													
33	RLC	C	:=	A ,A:=(A ....,A ;A)	0002									
34	RAL	C	:=	A ,A:=(A ....,A ;C)	0102									
35	RRC	A	:=	(A ,A ....,A):C:=A	0010									
36	RAR	A	:=	(C,A ....,A),C:A	0110									
	Sprung unbedingt				x: 0	1	2	3	4	5	6	7		
37	JMP	unbedingter Sprung			0404	0412	0504	0512	0604	0612	0704	0712	ladr	hadr
	Aufruf unbedingt				q: 0	1	2	3	4	5	6	7		
38	RMM	unbedingter Sprung an feste ADR mit Rueckkehr-absicht			0005	0013	0105	0113	0205	0213	0305	0313		
					x: 0	1	2	3	4	5	6	7		
39	CAL	UP-Sprung mit Rueckkehr-absicht			0406	0414	0506	0514	0606	0614	0706	0714	ladr	hadr
	Rueckkehr unbedingt				x: 0	1	2	3	4	5	6	7		
40	RET	unbedingter Ruecksprung zur letzten ADR im			0007	0015	0107	0115	0207	0215	0307	0315		

lfd.		Operationscodes					2.Byte	3.Byte
	Bedingte Operationen							
			o: C	Z	S	P		
		wenn : c = 0						
41	IFc	Sprung	0400	0408	0500	0508	ladr	hadr
42	CFc	Sprung mit Rueckkehrabs.	0402	0410	0502	0510	ladr	hadr
43	RFc	Rueckesprung zur letzten Adresse im Stapel	0300	0011	0103	0111		
		wenn : c = 1						
			c: C	Z	S	P		
44	ITc	Sprung	0600	0608	0700	0708	ladr	hadr
45	CTc	Sprung mit Rueckkehrabs.	0602	0610	0702	0710	ladr	hadr
46	RTc	Rueckesprung zur letzten Adresse im Stapel	0203	0211	0303	0311		
	Halt-Befehl							
47	HLT	Stop der ZVE	0000 0001 1515					
	Eingabe-Befehl							
48	INP	A : = <Eingabetor>						

Befehls-codierung	**Akkumulatorbits Bitdarstellung des Arbeitsregisters nach INP**							
	7	6	5	4	3	2	1	0
0401	AUB 5	= 1 :		Sel.4	Sel.3	Sel.2	Sel.1	
0403		Fehler-selbst-halte-kreis						
0405	Kursor-taste "Rueck-sprung"	Kursor-taste "Spalte"	Kursor-taste "Zeile"	Taste "Netz" aus				

	7	6	5	4	3	2	1	0
0407								
0409								
0411								
LBS 0413								
KMB		Schreib-nase	Ta "Loesch-Ausgabe-Sperre"	Bandan-fang - ende	KMB betriebs-bereit	Kassette eingelegt		
0415	Ta Sicht-barkeit "letztes Zeichen"	erstes Zeichen	Papier-kontrolle		letztes Zeichen	techn. Fehler		

Ausgabefehl

49 OUT Befehls-codierung 0500 — Ausgabetor : = < A >

Bitdarstellung des Arbeitsregisters vor OUT

# 0503								
# 0503	Markierg Schreiben	bit 7	Zeichen-speicher Sperren	bit 5	bit 4	bit 3	bit 2	bit 1
# 0505								

	7	6	5	4	3	2	1	0
0505	Code fuer Stelle bei alphanumerischer Anzeige							
0507	Programmanzeigelampen							
	La 8	La 4	La 2	La 1	Gueltig-keit			
0509	Fehlerlampe einschalter							
0511	Fehlerlampe loeschen							
0513	Setzen der Selektoren			Loeschen de: Selektoren				
	4	3	2	1	4	3	2	1
0515								
.	nur Bedeutung fuer Techniker							
.								
0715								

5.2. Ladebefehle (Transportbefehle)

Die Ladebefehle transportieren Daten zwischen den CPU-Registern und Speichern. Die Befehle muessen eine Ausgangsadresse, von der die Daten zu entnehmen sind und eine Zieladresse enthalten.
Der Inhalt des Quellenspeicherplatzes wird durch den Ladebefehl nicht veraendert. Beim Zielspeicherplatz wird das Ergebnis eingeschrieben.

5.2.1. Laden zwischen Arbeitsregistern

Hier handelt es sich um 1-Byte-Befehle. Die Verarbeitung in den Arbeitsregistern A, B, C, D, E, H, L (siehe Punkt 3.3.1.) erfolgt byteweise. Die Arbeitsregister dienen der Zwischenspeicherung beliebiger Informationen. Die Register H und L sind speziell als Adressregister fuer die Ansteuerung beliebiger Adressen (ROM und RAM) gedacht, sie koennen jedoch auch als normale Arbeitsregister Verwendung finden.

5.2.2. Laden mit "M"

Jeder beliebige Speicherplatz (M) im RAM und ROM wird durch H und L adressiert (siehe Punkt 3.3.2.1.).

5.2.3. Laden eines Direktoperanden "n"

Bei der Gruppe Ladebefehle gibt es eine Moeglichkeit, sowohl Arbeitsregister als auch "M" mit einem Direktoperanden (n) zu laden. Der Direktoperand ist dabei immer das dem Operationscode folgende Byte (siehe Punkt 3.3.3.).

5.3. Arithmetische Befehle und Vergleich

Dazu zaehlen die Additions- und Subtraktionsbefehle sowie der Inkrement- und der Dekrementbefehl.
Der Vergleich basiert ebenso wie die unter Punkt 5.4. beschriebenen logischen Operationen auf vorangegangenen arithmetischen Operationen.

5.3.1. Inkrement- und Dekrementbefehl (INr, DCr)

Der Inhalt der Arbeitsregister B, C, D, E, H, L (siehe Pkt. 3.3.1.) wird beim Inkrementbefehl (Dekrementbefehl) um 1 dual erhoeht (vermindert).
Das Arbeitsregister A kann **nicht** verwendet werden. Die Bedingungs-Flip-Flop's P, S, Z werden eingestellt.
Beim Inkrementieren eines Registers mit dem Inhalt 255 erhaelt man als Ergebnis Ø.
Beim Dekrementieren eines Registers mit dem Inhalt Ø erhaelt man als Ergebnis 255.

5.3.2. Additions- und Subtraktionsbefehle

Der Inhalt des Arbeitsregisters A wird entsprechend der Rechenregeln der Arithmetik mit einem Operanden dual verknuepft. Das Ergebnis steht im Arbeitsregister A.
Der Operand bleibt erhalten (Ausnahme: A wurde selbst als Operand verwendet).
Die Bedingungs-Flip-Flop's (C, Z, S, P) werden entsprechend der anliegenden Bedingungen eingestellt (siehe Punkt 2.2.6.).

5.3.2.1. Addition und Subtraktion eines Arbeitsregisters (ADr, ACr, SUr, SBr)

Als Operand wird ein Arbeitsregister (A, B, C, D, H, L) angegeben. Bei den Befehlen ACr bzw. SUr wird zusaetzlich ein vorangegangener Uebertrag beruecksichtigt und dual addiert bzw. subtrahiert (vgl. 3.1.).

5.3.2.2. Addition und Subtraktion eines Direktoperanden (ADn, ACn, SUn, SBM)

Als Operand wird ein Direktoperand (Konstante) angegeben. Bei den Befehlen ACn bzw. SBn wird zusaetzlich ein vorangegangener Uebertrag beruecksichtigt und dual addiert bzw. subtrahiert (vgl. 3.1.).

5.3.2.3. Addition und Subraktion eines Speicherplatzes M (ADM, ACM, SUM, SBM)

Als Operand wird ein durch die Register H und L adressierter Speicherplatz (M) im RAM oder ROM angegeben.
Bei den Befehlen ACM bzw. SBM wird zusaetzlich ein vorangegangener Uebertrag beruecksichtigt und dual addiert bzw. subtrahiert (vgl. 3.1.).

5.3.3. Vergleich

Die Bedingungs-Flip-Flop Z, C geben in der nachfolgend aufgefuehrten Weise Aufschluss ueber das Ergebnis.
Der Inhalt der Operanden bleibt erhalten.

CPr: Vergleiche A und r
CPM: Vergleiche A und M
CPn: Vergleiche A und n

Ergebnis:

	Bed.-Flip-Flop Z	C
(A) > (r), (M), (n)	0	0
(A) = (r), (M), (n)	1	0
(A) < (r), (M), (n)	0	1

5.4. Logische Operationen

Der Akkumulatorinhalt (A, 1. Operand) wird mit dem Inhalt eines anderen Registers (r), mit dem Inhalt (M) eines durch die Register H und L adressierten Speicherplatzes oder mit einer Konstanten (n, 2. Byte des Befehls), bitweise verknuepft. Das Ergebnis steht im Akkumulator.
Die Bedingungs-Flip-Flop's Z, S, P werden entsprechend dem Bitmuster des Ergebnisses aktualisiert.
Das C-Flip-Flop wird nicht gesetzt.

5.4.1. Konjunktion (UND)

Es erfolgt bitweise Verknuepfung der Operanden nach logischer UND-Funktion.

A: = A∧r/M/n

0	0	0
0	1	0
0	0	1
1	1	1

NDr: Konjunktion von A und r
(A): = (A)∧ (r)

NDM: Konjunktion von A und M
(A): = (A)∧ (M)

NDn: Konjunktion von A und n
(A): = (A)∧ (n)

5.4.2. Antivalenz (EXKLUSIVES ODER)

Es erfolgt bitweise Verknuepfung der Operanden nach logischer Funktion EXKLUSIVES ODER.

A : = (A) ⊕ r/M/n		
0	0	0
1	1	0
1	0	1
0	1	1

XRr: Antivalenz von A und r
(A): = (A) ⊕ (r)

XRM: Antivalenz von A und M
(A): = (A) ⊕ (M)

XRn: Antivalenz von A und n
(A): = (A) ⊕ (n)

5.4.3. Disjunktion (ODER)

Es erfolgt bitweise Verknuepfung der Operanden nach logischer Funktion ODER.

A : = A v r/M/n		
0	0	0
1	1	0
1	0	1
1	1	1

ORr: Disjunktion von A und r
(A) : = (A) v (r)

ORM: Disjunktion von A und M
(A) : = (A) v (M)

ORn: Disjunktion von A und n
(A) : = (A) v (n)

5.5. Sprungbefehle

5.5.1. Unbedingte Spruenge

5.5.1.1. Unbedingter Sprung (IMP)

Das Mikroprogramm wird bei dem als Zieladresse programmierten Befehl fortgesetzt.

5.5.1.2. Unbedingter Sprung an eine feste Adresse mit Rueckkehrabsicht (CAL)

Das Programm wird bei dem als ADR programmierten Befehl in einem mehrfach benoetigten Unterprogramm fortgesetzt.
Vor der Ausfuehrung dieser Sprungoperationen wird die Absprungadresse + 1 in Stapelregister in $F1_L$ und $F1_H$ zwischengespeichert.
In einem Unterprogramm usw. kann ein weiterer CAL-Sprung erfolgen, dabei wird der gesamte Stapelinhalt um einen Schritt nach unten verschoben. Das ist bis zu 7 x moeglich.

5.5.1.3. Unbedingter Sprung an eine feste Adresse mit Rueckkehrabsicht

Es erfolgt ein Sprung in eines der nachstehenden Unterprogramme:

Codierung	Unterprogramm
00/05	Vorwahlprogramm
00/13	Makrodecodierung
01/05	Warteprogramm
01/13	Tastaturabfrage
02/05	Druckkopfkontrolle
02/13	Registerrettung bei Tastaturabfrage
03/05	Protokollierprogramm
03/13	frei

Die Ansprungadresse + 1 wird in P1 L und P1 H zwischengespeichert.
Der gesamte Stapelinhalt wird um einen Schritt nach unten verschoben.

5.5.1.4. Unbedingter Sprung zur letzten ADR im Stapel (RET)

Dieser Befehl erfuellt die Funktion des unbedingten Ruecksprungs aus einem Unterprogramm.
Der Sprung erfolgt entsprechend der aktuellen Adresse in Stapelregister. Nach Ausfuehrung wird der Stapelinhalt um einen Schritt nach oben verschoben.

5.5.2. Bedingte Spruenge

Sprung zur Speicheradresse, wenn das Bedingungs-Flip-Flop c die Bedingung erfuellt. Andernfalls wird der naechste Befehl ausgefuehrt.
c steht fuer den Inhalt C, Z, S, P (vgl. 2.2.6.)

5.5.2.1. Sprung, wenn c = o (JFc)

5.5.2.2. Sprung von c = 1 (JTc)

5.5.2.3. Sprung mit Rueckkehrabsicht wenn c = 0 (CFc)

5.5.2.4. Sprung mit Rueckkehrabsicht wenn c = 1 (CTc)

5.5.2.5. Rueecksprung zur letzten ADR im Stapel wenn c = 0 (RFc)

5.5.2.6. Rueecksprung zur letzten ADR im Stapel wenn c = 1 (RTc)

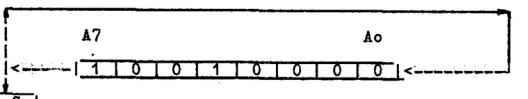
5.6. Sonstige Befehle

5.6.1. Rotationsbefehle

Vergleichbar mit Verschiebefehlen.
Rotationsbefehle haben die Aufgabe, Bitkombinationen innerhalb eines Bytes zu verschieben (z.B. beim Packen). Die Bitkombination muss im Arbeitsregister stehen.
Rotationsbefehle stellen das C-Flip-Flop (Uebertrag) ein.

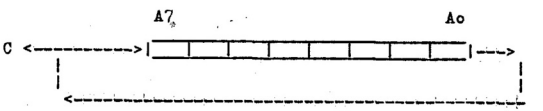
5.6.1.1. Linksrotation = RLC

Linksrotation in A
Bedingungs-Flip-Flop C schaltet parallel bei Uebertrag (A 7 geht nach A 0)



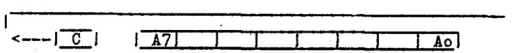
5.6.1.2. Rechnerrotation = RRC

Rechtsrotation in A
Bedingungs-Flip-Flop C schaltet parallel bei Uebertrag (Ao geht nach A 7)



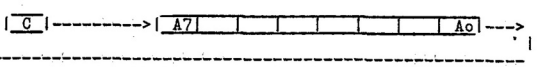
5.6.1.3. Linksrotation RAL

Linksrotation des Inhaltes von A bei Einschaltung des C-Flip-Flop als 9. Bit (A7 --> C, C --> Ao)



5.6.1.4. Rechtsrotation RAR

Rechtsrotation des Inhaltes von A bei Einschaltung des C-Flip-Flop als 9. Bit (C --> A7, Ao --> C)



5.6.2. Halt-Befehl HLT

Der HLT-Befehl ist ein maschineninterner Mikrobefehl, der Verwendung findet, wenn die Peripherie langsamer ist als die ZE.

5.7. Ein- und Ausgabebefehle

Ueber den Ein- und Ausgabebefehl werden die Ein- und Ausgabetore angesprochen.
Die Ein- und Ausgabetore stellen die Verbindung zwischen dem Mikroprozessor und der Peripherie dar.
(Belegung der Ein- und Ausgabetore siehe Pkt. 6. und Befehlsliste).

5.7.1. Befehl INP = Eingabebefehl

INP heisst: Lies den Inhalt des ausgewaehlten Eingabetores in das Arbeitsregister A.
Die Peripherie sendet ihre Daten- und Steuerinformationen ueber das angesteuerte Eingabetor in das Arbeitsregister A. Vom Programm kann anschliessend die Auswertung der belegten Bits im Arbeitsregister erfolgen.

5.7.2. Befehl OUT = Ausgabebefehl

OUT heisst: Schreibe den Inhalt des Arbeitsregisters A in das ausgewaehlte Ausgabetor.
Der Mikroprozessor sendet den Inhalt des Arbeitsregisters A (Daten- und Steuerinformationen) in das angesteuerte Ausgabetor. Die belegten Bits des Tores rufen die entsprechende Wirkung in der Peripherie hervor.

6. Belegung der Ein- und Ausgabetore

Es erfolgt nur die Beschreibung der fuer den Programmierer wichtigen Tore (manuelle Beeinflussung des Programmes, Anzeigen Lampen). Die Tore werden dabei durch ihre entsprechende Befehlscodierung gekennzeichnet (siehe Punkt 5.1.).

6.1. Eingabetore

0401 Dieses Tor dient der Selektorenabfrage.

Bit 4 = Selektor 4
Bit 3 = Selektor 3
Bit 2 = Selektor 2
Bit 1 = Selektor 1

Sind die Bits belegt, so wurde der entsprechende Selektor gesetzt.

Bedingung:
Vor dem Eingabebefehl muss das 5. Bit im Arbeitsregister mit 1 belegt werden, alle anderen Bits sind zu loeschen.

0403 Dieses Tor dient der Abfrage des Fehlerselbsthaltekreises

Bit 6 = Fehlerselbsthaltekreis

Ist dieses Bit belegt, so wurde die Fehlerlampe eingeschaltet.

0405 Dieses Tor dient zur Abfrage der Tastenbedienung

Bit 7 = Kursortaste "Rueecksprung"
Bit 6 = " "Zeile"
Bit 5 = " "Spalte"
Bit 4 = Taste "Netz aus"

Sind diese Bits belegt, dann wurden im Augenblick der Abarbeitung des Eingabebefehls die entsprechenden Tasten gedrueckt.

0413 Dieses Tor dient der Arbeit mit dem KMB.

KMB Bit 6 = Schreibnase

Ist dieses Bit belegt, befindet sich keine Schreibnase in der Kassette, die Kassette ist schreibgeschuetzt.

Bit 4 = Taste "Loeschen Ausgabesperre"

Ist dieses Bit belegt, wurde im Augenblick der Abarbeitung des Eingabebefehls diese Taste gedrueckt.

Bit 3 = Bandanfang - ende

Ist dieses Bit belegt, dann steht die Kassette auf dem Klarsichtband am Bandanfang oder -ende.

Bit 2 = KMB betriebsbereit

Ist dieses Bit belegt, so ist die Betriebsbereitschaft des KMB hergestellt.

Bit 1 = Kassette eingelegt

Ist dieses Bit belegt, so wurde eine Kassette eingelegt.

0415 Dieses Tor dient der Arbeit mit dem Drucker

Bit 7 = Taste Sichtbarkeit "Letztes Zeichen"

Ist dieses Bit belegt, so wurde im Augenblick der Abarbeitung des Eingabebefehls diese Taste gedrueckt.

Bit 6 = erstes Zeichen

Ist dieses Bit belegt, steht der Drucker in Grundstellung.

Bit 5 = Papierkontrolle

Ist dieses Bit belegt, befindet sich kein Papier im der Anlage.

Bit 2 = letztes Zeichen

Ist dieses Bit belegt, steht der Drucker in Endstellung.

Bit 1 = Technischer Fehler

Ist dieses Bit belegt, trat ein technischer Fehler am Drucker auf.

6.2. Ausgabetore

0503 Dieses Tor dient der Arbeit mit dem Bildschirm.
\# In Verbindung mit Tor 0505 kann es zum Loeschen des Bildschirms verwandt werden.

Bit 7 = Markierung schreiben

Ist dieses Bit belegt, so soll der Kursor ein "_" auf die im Tor 0505 angegebene Stelle des Bildschirmes schreiben.

Bit 6 = bit 7)
Bit 7 = bit 5) Spalte

Bit 3 = bit 4)
Bit 2 = bit 3) Zeile
Bit 1 = bit 2)
Bit 0 = bit 1)

In diesen Bits wird das Zeichen verschluesselt, welches auf der im Tor 0505 angegebenen Stelle auf dem Bildschirm erscheinen soll.

Bit 5 = Zeichenspeicher sperren

Ist dieses Bit belegt, dann kann nicht geschrieben werden.

0505 In den Bit 7-0 wird die Stelle angegeben, auf welcher
\# ein im Tor 0503 angegebenes Zeichen angezeigt werden soll.
Die Stelle kann 1-256 sein und wird dual in A angegeben.
1 = 0
256 = 1 1 1 1 1 1 1 1

0507 Dieses Tor dient der Arbeit mit den Programmanzeigelampen.

Bit 7 = Lampe 8
Bit 6 = Lampe 4
Bit 5 = Lampe 2
Bit 4 = Lampe 1

Bei Belegung dieser Bits koennen vom Mikroprogramm her die entsprechenden Programmanzeigelampen eingeschaltet werden.
Bit 3 = Gueltigkeit
Ist dieses Bit belegt, dann kann vom Programm her ein **Setzen** der Programmanzeigelampen erfolgen.

0509 Bei Aufruf dieses Tores erfolgt das Einschalten der Fehlerlampe.
Die Belegung von A ist dabei ohne Bedeutung.

0511 Bei Aufruf dieses Tores erfolgt das Ausschalten der Fehlerlampe.
Die Belegung von A ist dabei ohne Bedeutung.

0513 Dieses Tor dient der Arbeit mit den Selektoren.
Bit 7 = Selektor 4)
Bit 6 = Selektor 3)
Bit 5 = Selektor 2) Selektoren setzen
Bit 4 = Selektor 1)

Bit 3 = Selektor 4)
Bit 2 = Selektor 3)
Bit 1 = Selektor 2) Selektoren loeschen
Bit 0 = Selektor 1)
Bei Belegung dieser Bits werden die entsprechenden Selektoren gesetzt bzw. geloescht.

7. Uebersicht ueber die Belegung besonderer Speicherzeilen im RAM

Die Zeilen 00 bzw. 01 der RAM-Gruppe 0 werden vom ROM-Mikroprogramm benutzt fuer die Einstellung und Auswertung von Merkmalen.

7.1. Gruppe 0, Zeile 00

Die Speicherzeile 00 wird von den ROM-Programmteilen spezifisch belegt. Die Belegung der einzelnen Bits ist nur aus den Unterlagen des jeweiligen Mikroprogrammteiles ersichtlich.

7.2. Gruppe 0, Zeile 01

Diese Speicherzeile darf nur von speziellen Programmen fuer Zwischenspeicherungen genutzt werden.
Die Belegung der einzelnen Bits wird in der folgenden Uebersicht dargestellt und in den nachfolgenden Punkten naeher erlaeutert.

Uebersicht ueber die Belegung der einzelnen Bits

Byte \ bit	7	6	5	4	3	2	1	0
00	Makrobefehlszaehler (L-Teil)							
01	neue Gruppenadresse		Makrobefehlszaehler (H-Teil)					
02	Merkmal x	Sel x	Sel y	Sel A	Sel 3	Sel 2	Sel 1	Sel A
03	IR 3 Gr.-Adr.	IR2 Gr.-Adr.	IR1 Gr.-Adr.	IR0 Gr.-Adr.	EC minus	EC plus	EC ≠ 0	EC = 0
04	IR0 Bytesadresse							
05	IR1 Bytesadresse							
06	IR2 Bytesadresse							
07	IR3 Bytesadresse							
08	Lampe 8	Lampe 4	Lampe 2	Lampe 1				Konverter bit
09	x fuer Notstromfall reserviert							
10	x Merkmal "num. Anzeige auf ANA aendern"							
11	x Spalteninformation fuer Tastaturabfrage							
12	x Zeileninformation fuer Tastaturabfrage							
13	x Merkmal "y" (nach Tastaturabfrage 1. Teil)							
14	Merkmal x	Merkmal Start-taste		Sel 4	Sel 3	Sel 2	Sel 1	Sel A
15	Eingabezaehler: zaehlt die eingetasteten Ziffern und das Vorzeichen							

x) Die Bytes 09, 10, 11, 12 und 13 werden im Notstromfall mit anderen Informationen belegt.

7.2.1. 00-01-00

Dieses Byte enthaelt den L-Teil (niedere 8 bit) des Makrobefehlszaehlers. Dieser L-Teil gibt die aktuelle Byteadresse an.

7.2.2. 00-01-01

- bit 0-5 enthaelt den H-Teil (obere 6 bit) des Makrobefehlszaehlers.

- bit 6 und 7 enthaelt die neue Gruppenadresse fuer Gruppenumschaltungen.

bit	7	6	5	4	3	2	1	0
Gr. 0	0	0	1	0	0	0	0	0
Gr. 1	0	1	1	0	0	0	0	1
Gr. 2	1	0	1	0	0	0	1	0
Gr. 3	1	1	1	0	0	0	1	1

7.2.3. 00-01-02

Diese bits enthalten den Wert "1" wenn der entsprechende Selektor gesetzt ist.

7.2.4. 00-01-03

- bit 0-3 enthaelt eine "1", wenn der entsprechende Ergebniscod anliegt.

- bit 4-7 enthaelt die Gruppenadresse der Indexregister.

Die Arbeit mit den Indexgruppen kann nur mit der RAM-Gruppe 0 und 1 erfolgen, da fuer die Gruppendarstellung der IR nur je 1 bit zur Verfuegung steht.

7.2.5. 00-01-04

Auf diesem Byte wird die Byteadresse des IR0 abgespeichert. Die Adressierung der Indexregister benoetigt 8 bit fuer die Bytedarstellung, und fuer die Gruppendarstellung ist bei jedem Indexregister nur 1 bit vorgesehen.

7.2.6. 00-01-05

Byteadresse des IR 1
siehe Punkt 7.2.5.

7.2.7. 00-01-06

Byteadresse des IR 2
siehe Punkt 7.2.5.

7.2.8. 00-01-07

Byteadresse des IR 3
siehe Punkt 7.2.5.

7.2.9. 00-01-08

- Das "bit 0" wird durch den Befehl "KONV" auf "1" gestellt. Das VWP stellt DEG-Status (bit 0=0) ein.
- bit 1, 2, 3 ist nicht belegt.
- bit 4, 5, 6 und 7 sind entsprechend dem Lampencode gesetzt.

7.2.10. 00-01-09

Dieses Byte wird im Notstromfall belegt.

7.2.11. 00-01-10 bis 00-01-15

Diese Byte werden zur Zwischenspeicherung nach der Tastaturabfrage benoetigt.
Im Notstromfall haben diese Bytes eine besondere Belegung.

8. Funktionszuordnung im ROM

8.1. Allgemeine Hinweise

Der Gliederungspunkt gibt Auskunft ueber alle vorhandenen Mikroprogrammteile von robotron 1370 und 1255.
Die sich anschliessende Uebersicht enthaelt die Angaben, welche Mikroprogrammteile existieren und an welchen Adressen der ROM-Gruppen die verschiedenen Mikroprogrammteile stehen. Teile dieser Mikroprogramme koennen auch in Anwenderprogrammen genutzt werden. Dabei sind jedoch die Anfangs- und Anschlussbedingungen (Parameter) der einzelnen Mikroprogramme zu beachten sowie die Verschachtelung der Mikroprogrammteile.
Die scheinbaren Doppelbelegungen im Punkt 8.1.1. resultieren aus den verschiedenen Varianten der Anlage robotron 1370. Fuer die einzelnen Varianten ist der ROM unterschiedlich belegt, so dass bei der Benutzung dieser Mikroprogramme darauf geachtet werden muss, welches Mikroprogramm verwendet werden soll. Einige ausgewaehlte Mikroprogramme werden unter Punkt 8.2. naeher beschrieben.

8.1.1. Zuordnung zu ROM-Gruppen 1370

Kurzbezeichng. lt. techn. Unterlg.	Name des Mikroprogrammteiles	ROM-Adressen von	bis
	Grundausstattung:		
AL	1. Teil Vorwahlprogramm	00/00/00-	00/00/07
WD	Makrobef.-decodierung	00/00/08-	00/00/15
BH	1. Teil des UP-Verzoegerung durch Warteschleife	00/01/00-	00/01/05
YZ	Tastaturabfrage - Anfang -	00/01/08-	00/01/15
AS	Druckwerkskontrollprogr.	00/02/00-	00/02/06
UN	UP num. Tastaturabfrage mit Registerrettung	00/02/08-	00/02/15
XE	Tastaturmatrixabfrage	00/03/00-	00/03/10
BR	Speicherplatz mit Codezeichen Laden (ohne Weiterzaehlen ADR)	00/03/11-	00/04/04
BN	Speicherplatz mit Codezeichen Laden (mit Weiterzaehlen ADR)	00/04/05-	00/04/13
WA	Arithmetische Befehle Addition	00/04/14-	00/05/11
WU	Sonstige Befehle - Gruppenumschaltung	00/05/12-	00/06/09
BV	Abwaertszaehlen Indexregister	00/06/10-	00/07/10
AV	Fortsetzung der Makrobefehlscodierung	00/07/11-	00/13/05
UL	UP Loeschen	00/13/06-	00/14/01
UM	UP Makrobef.-zaehler weiterzaehlen	00/14/02-	00/15/07
UZ	UP weiterzaehlen ADR	00/15/08-	01/00/03
UU	UP Nulltest	01/00/04-	01/01/04
WS	Arithmetische Befehle - Subtraktion	01/01/05-	01/01/08
CZ	keine Operation - Loeschen Ergebniscode (EC = 0)	01/01/09-	01/02/05
CR	Laden des IR mit ADR in Gruppe 0	01/02/06-	01/02/13
CN	Laden des IR mit ADR in Gruppe 1	01/02/14-	01/04/03
CV	Sprung ins Vorwahlprogramm	01/04/04-	01/15/05
AU	Tastaturabfrage	01/04/05-	01/15/05
XC	Tastaturmatrixspaltenabfrage	01/15/06-	02/03/07
DR	Zeichentransport von R (ohne Weiterzaehlen ADR)	02/03/14-	02/07/11
DN	Zeichentransport von R (mit Weiterzaehlen ADR	02/07/12-	02/08/03
WP	Arithmetische Befehle - Addition 1	02/08/04-	02/09/11
CD	2. Teil Vorwahlprogramm	02/09/12-	02/14/15
AX	Zahlenpruefung	02/15/00-	03/03/07
EN	Zeichentransport nach R (mit Weiterzaehlen d. ADR)	03/03/08-	03/03/15
ER	Zeichentransport nach R (ohne Weiterzaehlen d. ADR)	03/04/00-	03/05/04
WM	Arithmetische Befehle - Subtraktion 1	03/05/05-	03/05/09
WX	Tabelle fuer Makrobefehlsdecod.	03/05/10-	03/10/03
CB	UP Programmeingabe manuell	03/10/04-	03/14/03
UV	UP Vorzeichenaufbereitung	03/14/04-	03/14/13

UK	UP Addition	03/14/14-04/02/07
WT	Arithmetische Befehle - Transport	04/02/08-04/03/10
WI	Sprungbefehle	04/03/11-04/07/10
US	UP Subtraktion	04/07/11-04/11/00
UW	UP Wortlaenge	04/11/01-04/12/10
UA	UP Addition	04/12/11-04/14/02
AQ	Makroprogramm im P ROM	04/14/03-04/15/03
AP	Unterprogramm fuer eine Wartezeit von 4,5 ms	04/15/04-04/15/11
UE	UP E-L-Aktualisierung	04/15/12-05/00/09
WL	Arithmetische Befehle - Loeschen	05/00/10-05/01/08
PZ	Netzabschaltung	05/01/09-05/02/05
FY	Setzen des Selektors X	05/02/06-05/02/11
Spr. nach FQ --->		05/02/12-05/03/01
FW	Zeilenschaltung einzeilig vorwaerts	05/03/02-05/03/11
FV	Formularvorschub bis zum progr. Stop	05/03/12-05/04/00
FU	Formularvorschub bis zur naechsten Seite	05/04/01-05/04/05
FT	Loeschen der Bildschirmanzeige	05/04/06-05/05/06
FS	Setzen Selektor Y	05/05/07-05/05/10
FR	Lampen setzen, Selektor 1, XY, Loeschen, Komma fuer num. Anzeige	05/05/11-05/10/04
FK	Steuerbefehle - num. Ergebnisanzeige	05/10/05-05/11/04
FJ	Sprung ins Mikroprogramm	05/11/05-05/11/11
Spr. nach <------		05/11/12-05/11/14
FP		05/11/15-06/01/14
CA	Programmeingabe manuell	06/01/15-06/02/15
UO	UP Subtraktion, komplett	06/03/00-06/04/01
UC	UP Ergebniscode - Aufbereitung	06/04/02-06/15/15
HZ	Wortweise Eingabe	07/00/00-07/06/12
HR	Wortweise Ausgabe OWd 8...15	07/06/13-07/10/06
WV	Arithmetische Befehle - Verschiebung	07/10/07-07/10/09
Spr. nach HZ --->		07/10/10-09/01/13
GW	Eingabe ueber Alphatastatur	09/01/14-09/05/09
XA	num. LED-Anzeige	09/05/10-09/06/08
ZB	Tabelle fuer num. LED-Anzeige	09/06/09-09/09/09
GN	Ausdruck eines Speicherbereiches	09/09/10-09/12/08
HW	Loechen eines Zeichens	09/12/09-10/01/10
HS	Musterdruck zugehoerig zur wortweisen Ausg.	10/01/11-10/05/00
HT	UP zu Druckprogrammen	10/05/01-10/07/06
FQ	Druckkopfbewegung rechts/links	10/07/00-10/04/15
GR	Druck eines Codierzeichens	10/08/00-10/14/15
AZ	Tastaturtabelle lat. (Standard)	10/15/00-11/00/13
HY	Sichtbarmachung des letzten Druckzeichens	

Alpha tast. / kein Drucker Tastatur- / Bild- tab.f.lat. (Stand.) / schirm

HU	UP zur Lochbandeingabe	11/00/14-11/11/09	Lochbandstanser
HX	Lochbandausgabe zugehoerig zur wortweisen Ausgabe	11/11/10-11/14/00	
WC	Arithmetische Befehle - Vergleich	12/01/13-12/03/03	
Spr. nach GR	-->	12/03/04-12/03/06	
Spr. nach GN	-->	12/03/07-12/03/09	
Spr. nach GX	-->	12/03/10-12/03/12	
Spr. nach GW	-->	12/03/13-12/04/02	
GL	Ausgabe eines Speicherbereiches zum Lochbandstanzer	12/04/03-12/06/06	
GQ	Lochen eines Codierzeichens	12/06/07-12/06/13	
ZF	Setzen und Loeschen der Selektoren 1, 2, 3 und 4	12/11/02-12/12/04	
ZB	Anfangsloeschung der Bildschirmanzeige	12/12/10-12/13/05	
HV	Sendercode im ROM fuer Lochbandstanser	13/00/00-13/15/15	Progr.-ein-u. aus. von bsw. auf Karte
UC	Programmierung von der Karte	14/00/00-14/05/12	
CB	Programmeingabe auf Karte	14/05/13-14/15/11	
GW (teilweise)	Eingabe ueber Alphatastatur	07/10/10-09/01/13	
AC	zum Anzeige auf Bildschirm	09/01/14-09/03/15	mit Bildschirm
GX	Anzeige eines Speicherbereiches auf Bildschirm	09/04/00-09/05/15	
GN	Ausdruck eines Speicherbereiches	09/06/09-09/09/09	
HW	Lochen eines Zeichens	09/09/10/09/12/08	
HS	Musterdruck zugehoerig zur wortweisen Ausgb	09/12/09-10/09/10	
HT	UP zu Druckprogrammen	10/01/11-10/05/00	Drucker, Tastatur-tab. kyrill. (SU)
FQ	Druckkopfbewegung rechts/links	10/05/01-10/07/06	
GR	Druck eines Codierzeichens	10/07/07-10/07/15	
QA	Tastaturtabelle kyrillisch (SU)	10/08/00-10/14/15	
HY	Sichtbarmachung des letzten Druckzeichens	10/15/00-11/00/13	
HU (teilweise)	UP zur Lochbandausgabe	11/00/14-11/11/09	
HP (teilweise)	Gemeinsame UPMBKG	11/03/00-12/01/12	gemeinsame KMB-Programme
WC	Arithmetische Befehle - Vergleich	12/01/13-12/03/03	
Spr. nach GR	-->	12/03/04-12/03/06	
Spr. nach GN	-->	12/03/07-12/03/09	
Spr. nach GX	-->	12/03/10-12/03/12	
Spr. nach GW	-->	12/03/13-12/04/02	

BE	Wortweise Ausgabe Inhalt Arbeitsspeicher nach Datentraeger	12/04/03-12/09/11	
GU	Befehl zum Einlesen eines Datenblocks vom KMBG auf Speicher	12/09/14-12/11/01	
ZF	Setzen und Loeschen der Selektoren 1, 2, 3 und 4	12/11/02-12/12/04	
ZE	Anfangsloeschung der Bildschirmanz.	12/12/10-12/13/05	
Spr. nach ZF -->		12/13/06-12/13/08	
FP	Masch.Steuerbef. KMBG-Decodierung	12/13/09-13/03/09	Pk1
BG	Spezielle UP fuer KMBG Pk1	13/03/10-13/11/02	Pk1
CO (teilweise)	CRC-Verrechnung fuer KMBG	13/11/03-13/14/03	
CH	Umspulprogramm fuer KMG	13/14/04-13/15/02	KMBG
BF (teilweise)	Gemeinsame UP MBKG	13/15/03-13/15/15	KMBG
CF	Programmeingabe von KMBG	14/00/00-14/04/03	
CO (teilweise)	CRC-Verrechnung fuer KMGB	14/04/04-14/05/12	
CG	Progr.-ausgabe auf KMBG	14/05/13-14/14/11	
CO (teilweise)	CRC-Verrechnung fuer KMBG	14/14/12-14/15/15	
ZD	Notstrom	15/00/00-15/15/13	

8.1.2. Uebersicht Funktionszuordnung zu den ROM-Gruppen 1255

Kurzbezeichng. Unterlg.	Name des Mikroprogrammteils	ROM-Adressen von bis
AL	1. Teil Vorwahlprogramm	0/00/00/00-0/00/00/07
WD	Makrobefehlscodierung (RST 1)	0/00/00/08-0/00/00/15
BH	1. Teil UP Verzoegerung durch Warteschleife (RST 2)	0/00/01/00-0/00/01/07
YZ	Tastaturabfrage (RST 3)	0/00/01/08-0/00/01/15
AS	Druckwerkskontrollprogramm	0/00/02/00-0/00/02/07
UN	UP num. Tastaturabfrage m. Registerrettung (RST 5)	0/00/02/08-0/00/02/15
XE	Tastaturabfrage 1. Fortsetzung	0/00/03/00-0/00/03/10
WA	Arithmetische Befehls Addition	0/00/03/11-0/00/04/08
WU	Sonstige Befehle Gruppenumschaltung DEG	0/00/04/09-0/00/05/06
BV	Abwaertszaehlen des IR/DEG	0/00/05/07-0/00/06/02
ZT	Speicherplatz m. Codezeichen laden	0/00/06/03-0/00/10/12
AV	Fortsetzung Makrobefehlsdecodierung	0/00/10/13-0/01/00/11
WS	Arithmetische Befehle Subtraktion	0/01/00/12-0/01/00/15
NZ	keine Operation, Loe Merkmal EC=0, Rueckspr. aus UP	0/01/01/00-0/01/03/04
NV	Sprung in VWP; Setzen Merkmal DEG/Konv.	0/01/03/05-0/01/04/01

CR	Laden d. IR Adresse in Gr. 0 (DEG) KONV.F.GW	0/01/04/02-0/01/04/09
CN	Laden d. IR m. Adresse in Gr. 1 (DEG) KONV.F.GW	0/01/04/10-0/01/05/15
UL	UL Loeschen	0/01/06/00-0/01/06/11
UM	UP Makrobefehlszaehler weiterzaehlen	0/01/06/12-0/01/08/01
UZ	UP Weiterzaehlen Adresse	0/01/08/02/0/01/08/13
UU	UP Nulltest	0/01/08/14-0/01/09/14
UV	UP Vorzeichenaufbereitung	0/01/09/15-0/01/10/08
UK	UP Addition	0/01/10/09-0/01/14/02
US	UP Subtraktion	0/01/14/03-0/02/01/08
WP	Arithmetischen Befehle Addition 1	0/02/01/09-0/02/03/00
DZ	Laden IR f. Konverter, direkter Steuerbefehl	0/02/03/01-0/02/07/02
TV	Transport Zeichen von AR	0/02/07/03-0/02/13/14
UW	UP Wortlaenge	0/02/13/15-0/02/15/08
UA	UP Addition	0/02/15/09-0/03/01/00
WM	Arithmetische Befehle Subtraktion	0/03/01/01-0/03/01/05
EZ	Addition/Subtraktion 1, Beginn Blocktransport, Zeichenvergl.	0/03/01/06-0/03/04/04
AR	Adressrechnung	0/03/04/05-0/03/08/07
TN	Transport Zeichen nach A	0/03/08/08-0/03/12/03
AF	UP .. Wartezeit v. 4,5 ms	0/03/12/04-0/03/12/11
UE	UP E-, L Aktualisierung	0/03/12/12-0/03/13/09
XC	Tastaturabfrage 1.Teil Fortsetzung/Schluss	0/03/13/10-0/04/01/11
WT	Arithmetische Befehle Transport ADR --> R	0/04/01/12-0/04/02/14
SP	Sprung	0/04/02/15-0/04/10/02
BT	Blocktransport	0/04/10/03-0/04/13/01
WX	Tabelle f. Makrobefehlsdecodierung	0/04/13/02-0/05/01/11
WL	Arithmetische Befehle Loeschen	0/05/01/12-0/05/02/10
FZ	Netzabschaltung	0/05/02/11-0/05/03/07
FY	Setzen Selektor X	0/05/03/08-0/05/03/13
FW	Zeilenschaltung einzeilig vorwaerts	0/05/03/14-0/05/04/07
FV	Formularvorschub bis zum programmierten Stop	0/05/04/08-0/05/04/12
FU	Formularvorschub bis zur naechsten Seite	0/05/04/13-0/05/05/01
FS	Setzen Selektor Y	0/05/05/02-0/05/05/05
FR	Lampen Setzen, Sel. 1,X,Y, Loeschen, Komma f. num.Anzeige	0/05/05/06-0/05/09/15
FQ	Druckkopfbewegung rechts/links	0/05/10/00-0/05/12/05

Sprung an FP

ZP	Setzen und Loeschen der Selektoren 1-4	0/05/12/09-0/05/13/11
FK	Numerische Anzeige	0/05/13/12-0/05/14/11
FJ	Sprung ins Mikroprogramm	0/05/14/12-0/05/15/02
FO	Steuerbefehle Magnetband	0/05/15/03-0/06/03/11
ZP	Befehl Zahlenpruefung	0/06/03/12-0/06/12/13
CW	Codewandel	0/06/12/14-0/07/00/00
DS	Aus- und Eingabe Magnetband	0/07/00/08-0/07/03/07
WV	Arithmetische Befehle Verschiebung	0/07/03/08-0/07/07/01
HR	Wortweise Ausgabe OW 0...15	0/07/07/02-0/07/13/14
HZ	Wortweise Eingabe	0/07/13/15-0/08/09/12
VE	Befehl Zeichenvergleich	0/08/09/13-0/08/13/09
AU	Tastaturabfrage Fortsetzung/ Schluss	0/08/13/10-0/09/08/10
CA	Progr.-eingabe manuell	0/09/08/11-0/09/14/10
CB	UP Progr.-eingabe manuell	0/09/14/11-0/10/02/10
AZ	Tastaturtabelle lat. (Stand.)	0/10/02/11-0/10/09/10
UO	UP Subtraktion komplett	0/10/09/11-0/10/10/11
UC	UP Ergebniscodeaufbereitung	0/10/10/12-0/10/11/13
HU	UP zur LB-Ausgabe	0/10/11/14-0/10/13/15
HY	Sichtbarmachung d. letzten Druckzeichens	0/10/14/00-0/10/15/13
HS	Musterdruck zugehoerig zur wortweisen Ausgabe	0/10/15/14-0/11/04/15
HT	UP zu Druckprogrammen	0/11/05/00-0/11/08/05
XA	Numerische LED-Anzeige	0/11/08/06-0/11/12/01
ZB	Tabelle f. num. LED-Anzeige	0/11/12/02-0/11/13/00
CD	2. Teil VWP	0/11/13/01-0/12/02/04
WC	Arithmetischer Befehl Vergl.	0/12/02/05-0/12/03/11
GR	Druck eines Codezeichens	0/12/03/12-0/12/04/04
KN	Ausdruck eines Speicherber.	0/12/04/05-0/12/08/15
	Sprung an GU	0/12/09/00-0/12/09/02
	Sprung an BE	0/12/09/03-0/12/09/05
GW	Eingabe ueber Alphatastatur	0/12/09/06-0/14/00/09
FP	Steuerbefehl KMBG	0/14/00/10-0/14/09/15
GU	Lesen KMBG	0/14/10/00-0/15/05/14
BE	Ausgabe von Daten auf KMBG	0/15/05/15-1/00/00/04
CG	Progr.-ausgabe auf KMBG	1/00/00/05-1/00/10/02
CF	Progr.-eingabe vom KMBG	1/00/10/03-1/00/13/10
BG	UP f. KMBG	1/00/13/11-1/01/12/04
BF	UP BF f. KMBG (Schreiben)	1/01/12/05-1/03/01/09
CO	CRC-Zeichen-Berechnung	1/03/01/10-1/03/07/05
CH	Umspulprogramm f. KMBG	1/03/07/06-1/03/09/02

8.2. Ausgewaehlte Mikroprogrammteile 1370

Programmeingabe von PK:

```
Programm  C C
ROM 14 Byte 0000
Ansprung durch C A L
nach  P E und T+ im VWP
```

Verriegeln einer MBK:

```
Programm B F
ROM 11 Byte 1209
Ansprung durch C A L
anschliessend muss LAI 0006
                   1413
                 CUT 0615
               RST 1 0013 programmiert
                          werden
```

Warteprogramm von 4,5 ms:

```
Programm AF,A
ROM 04  Byte 1504
Ansprung durch CAL
anschliessend muss RST 1 0013 programmiert
                              werden
```

Zahlenpruefung:

```
Programm AX, ZP
ROM 02 Byte 1500
Ansprung durch CAL
vorher muss LLI 0306
                0000
            LMI 0314
                Nr. d. Eingabetabelle fuer
                Fehlerwiederholung
```

Rueckspulen ohne EM-Aufzeichnung:

```
Programm BF
ROM 11 Byte 0808
Ansprung durch CAL
vorher muss LAI 0006 programmiert werden
            0201
nach UP-Abarbeitung muss RST 1 programmiert
werden
```

Verspulen ohne EM-Aufzeichnung:

```
Programm BF
ROM 11 Byte 0808
Ansprung durch CAL
vorher muss LAI 0006

nach UP-Abarbeitung muss RST 1 programmiert
werden.
```

Loeschen des Rechenregisters:	**Programm UL** **ROM 00 Byte 1308** **Ansprung durch CAL** **vorher muss LEI 0206** **0015 (15 Stw.) programmiert werden** **LLI 0306 programmiert werden** **0500** **nach UP-Abarbeitung muss RST 1 programmiert werden**

9. Hilfsprogramme bei der Arbeit mit Mikroprogrammen

Die nachfolgend aufgefuehrten Hilfsprogramme sind Neuererleistungen und sind auf der Basis von Nachnutzungsvertraegen verfuegbar.

9.1. Testhilfe TEMIK

Kurzbeschreibung

TEMIK ist ein Testhilfsprogramm zum Test von Programmen bzw. Programmteilen, die in der Maschinensprache des Mikroprozessors U808 bzw. INTEL 8008 geschrieben sind (Mikroprgramme). Durch TEMIK werden folgende Funktionen realisiert:

1. Schrittweise Abarbeitung des zu testenden Programmes mit Druck und Stop nach jedem Schritt
2. Schrittweise Abarbeitung des zu testenden Programmes mit Druck und Stop auf einer vorgegebenen Befehlsadresse
3. Abarbeitung des zu testenden Programmes ohne Druck mit Stop auf einer vorgegebenen Befehlsadresse
4. Nach jedem Stop sind moeglich:
   - beliebiger Wechsel der Abarbeitungsart
   - Beginn mit neuer Anfangsadresse
   - Veraenderung von Speicherinhalten (RAM-Inhalt, CPU-Register)
   - Veraenderung der Flag-Einstellung, der Stellung des Stack-Pointers und des Stack-Inhaltes
   - visuelle Kontrolle (tetradendezimale numerische Anzeige) von Speicherinhalten (RAM, ROM, CPU)

Damit kann eine dynamische Programmtestung bei einfacher Bedienung durchgefuehrt werden.

Bei Abarbeitung mit Druck erfolgt die Ausgabe je Abarbeitungsschritt von

- Befehlsadresse (RAM, ROM)
- Befehlsmnemonik Befehlscodierung
- Inhalt der CPU-Register A, B, C, D, E, H, L nach Abarbeitung des Befehls
- Inhalt des durch H L adressierten Speicherplatzes (M)
- Flag-Einstellung nach Abarbeitung des Befehls

Die zu testenden Programme koennen im RAM (00/0000..09/1115) oder im ROM (gesamter Speicherbereich) des Magnetbandkonverters robotron 1255 gespeichert sein.

9.2. Dokumentation von Mikroprogrammteilen DOKMIK

Das Programm Dokmik ermoeglicht die maschinelle Dokumentierung von Programmen bzw. Programmteilen, die in der Maschinensprache des Mikroprozessors U808 bzw. Intel 8008 geschrieben sind und sich im RAM (00/0600..10/1515) oder ROM (gesamter Speicherbereich) des Magnetbandkonverters robotron 1255 befinden.
Die Dokumentierung erfolgt in der Form

Befehlsmnemonik - Befehlscode - Befehlsadresse

Beispiel:

```
INH  0208  02/0000
LHI  0214  02/0001
-    0203  02/0002
```

Bei 2- und 3-Byte-Befehlen wird fuer das zweite bzw. dritte Byte in der Spalte "Befehlsmnemonik" das Minus-Zeichen (-) gedruckt. Ebenso werden Konstantenbereiche dokumentiert.
Mit DOKMIK kann auch eine Dokumentation in der Form

Speicherinhalt - Speicheradresse

erfolgen (z.B. Dokumentieren von Programmen ohne Mnemonik).

Beispiele:

```
0906 03/1214
1311 03/1215
0001 03/1300
```

Der Druck erfolgt seitenweise. Die Zeilenanzahl/Seite und der Abstand zwischen zwei Druckseiten sind beliebig vorgebbar. Der Druck kann fortlaufend oder mit Stop nach jeder Seite erfolgen. Die Abstaende zwischen den Druckspalten sind ebenfalls veraenderbar.

00	HLT	HLT	RLC	RFC	ADI	RST	LAI	RET	INB	DCB	RRC	RFZ	ACI	RST	LBI	RET
01	INC	DCC	RAL	RFS	SUI	RST	LCI		IND	DCD	RAR	RFP	SBI	RST	LDI	
02	INE	DCE		RTC	NDI	RST	LEI		INH	DCH		RTZ	XRI	RST	LHI	
03	INL	DCL		RTS	ORI	RST	LLI					RTP	CPI	RST	LMI	
04	JFC	INP	CFC	INP	JMP	INP	CAL	INP	JEZ	INP	CFZ	INP	JMP	INP	CAL	INP
05	JFS	OUT	CFS	OUT		OUT		OUT	JFP	OUT	CFP	OUT		OUT		OUT
06	JTC	OUT	CTC	OUT		OUT		OUT	JTZ	OUT	CTZ	OUT		OUT		OUT
07	JTS	OUT	CTS	OUT		OUT		OUT	JTP	OUT	CTP	OUT		OUT		OUT
08	ADA	ADB	ADC	ADD	ADE	ADH	ADL	ADM	ACA	ACB	ACC	ACD	ACE	ACH	ACL	ACM
09	SUA	SUB	SUC	SUD	SUE	SUH	SUL	SUM	SBA	SBB	SBC	SBD	SBE	SBH	SBL	SBM
10	NDA	NDB	NDC	NDD	NDE	NDH	NDL	NDM	XRA	XRB	XRC	XRD	XRE	XRH	XRL	XRM
11	ORA	ORB	ORC	ORD	ORE	ORH	ORL	ORM	CPA	CPB	CPC	CPD	CPE	CPH	CPL	CPM
12	LAA	LAB	LAC	LAD	LAE	LAH	LAL	LAM	LBA	LBB	LBC	LBD	LBE	LBH	LBL	LBM
13	LCA	LCB	LCC	LCD	LCE	LCH	LCL	LCM	LDA	LDB	LDC	LDD	LDE	LDH	LDL	LDM
14	LEA	LEB	LEC	LED	LEE	LEH	LEL	LEM	LHA	LHB	LHC	LHD	LHE	LHH	LHL	LHM
15	LLA	LLB	LLC	LLD	LLE	LLH	LLL	LLM	LMA	LMB	LMC	LMD	LME	LMH	LML	HLT
	/00	/01	/02	/03	/04	/05	/06	/07	/08	/09	/10	/11	/12	/13	/14	/15

Erklaerung:

Additions-u. Subtraktionsbefehle — AD./Su. ohne Beruecks. eines UE
AC./SB. mit Beruecks. eines Uebertrages

Ein- und Ausgabebefehle

Inkrementbefehle

Ladebefehle

Logische Operationen — N.. Und
O.. Oder
X.. Antivalenz
C.. Vergleich

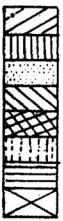
Rotationsbefehle

Sprungbefehle — 2,3-Mehrbyte-Befehle
x-Codierung innerh. Umrandung
beliebig; Schraffur-Vorzugscode

Haltebefehle

00	RET	RET	RLC	RFC	ADI$_2$	RST$_0$	LAI$_2$	RET$_X$	INB	DCB	RRC	RFZ	ACI$_2$	RST$_1$	LBI$_2$	RET$_X$
01	INC	DCC	RAL	RFS	SUI$_2$	RST$_2$	LCI$_2$		IND	DCD	RAR	RFP	SBI$_2$	RST$_3$	LDI$_2$	
02	INE	DCE		RTC	NDI$_2$	RST$_4$	LEI$_2$		INH	DCH		RTZ	XRI$_2$	RST$_5$	LHI$_2$	
03	INL	DCL		RTS	ORI$_2$	RST$_6$	LLI$_2$					RTP	CPI$_2$	RST$_7$	LMI$_2$	
04	JFC$_3$	INP	CFC$_3$	INP	JMP$_3$	INP	CAL$_3$	INP	JFZ$_3$	INP	CFZ$_3$	INP	JMP$_3$	INP	CAL$_3$	INP
05	JFS$_3$	OUT	CFS$_3$	OUT		OUT		OUT	JFP$_3$	OUT	CFP$_3$	OUT		OUT		OUT
06	JTC$_3$	OUT	CTC$_3$	OUT		OUT		OUT	JTZ$_3$	OUT	CTZ$_3$	OUT		OUT		OUT
07	JTS$_3$	OUT	CTS$_3$	OUT		OUT		OUT	JTP$_3$	OUT	CTP$_3$	OUT		OUT		OUT
08	ADA	ADB	ADC	ADD	ADE	ADH	ADL	ADM	ACA	ACB	ACC	ACD	ACE	ACH	ACL	ACM
09	SUA	SUB	SUC	SUD	SUE	SUH	SUL	SUM	SBA	SBB	SBC	SBD	SBE	SBH	SBL	SBM
10	NRA	NDB	NDC	NDD	NDE	NDH	NDL	NDM	XRA	XRB	XRC	XRD	XRE	XRH	XRL	XRM
11	ORA	ORB	ORC	ORD	ORE	ORH	ORL	ORM	CPA	CPB	CPC	CPD	CPE	CPH	CPL	CPM
12	LAA	LAB	LAC	LAD	LAE	LAH	LAL	LAM	LBA	LBB	LBC	LBD	LBE	LBH	LBL	LBM
13	LCA	LCB	LCC	LCD	LCE	LCH	LCL	LCM	LDA	LDB	LDC	LDD	LDE	LDH	LDL	LDM
14	LEA	LEB	LEC	LED	LEE	LEH	LEL	LEM	LHA	LHB	LHC	LHD	LHE	LHH	LHL	LHM
15	LLA	LLB	LLC	LLD	LLE	LLH	LLL	LLM	LMA	LMB	LMC	LMD	LME	LMH	LML	HLT
	/00	/01	/02	/03	/04	/05	/06	/07	/08	/09	/10	/11	/12	/13	/14	/15

Erklärung:



Additions- u. Subtraktionsbefehle — AD./SU. ohne Berücks. eines Ü; AC./SB. mit " eines Übertr.

Ein- u. Ausgabebefehle

Inkrementbefehle

Ladebefehle

Logische Operationen — N.. Und; O.. Oder; X.. Antivalenz; C.. Vergleich

Rotationsbefehle

Sprungbefehle

Haltbefehle

2,3- Mehrbyte-Befehle

x- Codierung innerh. Umrandung beliebig; Schraffur=Vorzugscod